L2: Entry 1 of 1 File: USPT Jan 6, 2004

DOCUMENT-IDENTIFIER: US 6675261 B2
TITLE: Request based caching of data store data

Abstract Text (1):
A request, such as those embedded in URLs and XML documents, is assigned to a thread of execution in a server that is in communication with a data store. The thread of execution includes a thread local storage with a pointer to a cache object. The cache object maintains copies of data store entries frequently accessed by the assigned request. The cache object is accessed in response to data store access commands arising from the request. When a data store access command specifies a data store entry not found in the cache object, the server creates and loads a corresponding cache object entry. The cache object is not updated when other requests alter data store entries, and memory access commands arising from other requests cannot cause the cache object to be accessed. When the request causes the server to write data to the data store, the cache object also maintains a copy of the written data. The server retrieves the written data from the cache object in response to subsequent data store access queries arising from the request. The cache object is destroyed once the server completes a response to the request.

Brief Summary Text (7):
Identity Systems have become more popular with the growth of the Internet and the use of networks and other information technologies. In general, an Identity System provides for the creation, removal, editing and other management of identity information stored in various types of data stores. The identity information pertains to users, groups, organizations and/or things. For each entry in the data store, a set of attributes is stored. For example, the attributes stored for a user may include a name, address, employee number, telephone number, email address, user ID and password. The Identity System can also manage access privileges that govern the subject matter an entity can view, create, modify or use in the Identity System.

Brief Summary Text (13):
In one embodiment of the present invention, an Identity Server receiving a request interprets a command in the request to call for an access to a data store entry. In response to the command, the Identity Server accesses a first entry in the cache object that corresponds to the data store entry called for in the command. If the cache object does not include an entry corresponding to the data store entry, the Identity Server creates and loads a corresponding entry in the cache object.

Drawing Description Text (17):
FIG. 16 is a flow chart describing an overview of a process for creating and using workflows.

Drawing Description Text (18):
FIG. 17 is a flow chart describing one embodiment of a process for creating a template.

Drawing Description Text (19):
FIG. 18 is a flow chart describing one embodiment of a process for creating a workflow.

Drawing Description Text (23):
FIG. 22 is a flow chart describing one embodiment of a process for creating a cross application workflow.

Drawing Description Text (64):
FIG. 60 is a flow chart describing a process for creating a policy domain.

Drawing Description Text (68):
FIG. 64 is a flow chart describing a process for creating a policy.

Detailed Description Text (11):
The Identity System includes Web Pass 38, Identity Server 40 and Directory Server 36. Identity Server 40 manages identity profiles. An identity profile is a set of information associated with a particular entity (e.g. user, group, organization, etc.). The data elements of the identity profile are called attributes, which are discussed in more detail below. An attribute may include a name, value and access criteria. The Identity Server includes three main applications, which effectively handle the identity profiles and privileges of the user population: User Manager 42, Group Manager 44, and Organization Manager 46. User Manager 42 manages the identity profiles for individual users. Group Manager 44 manages identity profiles for groups. Organization Manager 46 manages identity profiles for organizations. Identity Server 40 also includes Publisher 48, an application that enables entities to quickly locate and graphically view information stored by Directory Server 36. In one embodiment, Web Pass 38 is a Web Server plug-in that sends information back and forth between Identity Server 40 and the Web Server 20, creating a three-tier architecture. The Identity System also provides a Certificate Processing Server (not shown in FIG. 1) for managing digital certificates.

Detailed Description Text (12):
User Manager 42 handles the functions related to user identities and access privileges, including creation and deletion of user identity profiles, modification of user identity profile data, determination of access privileges, and credentials management of both passwords and digital certificates. With User Manager 42, the create, delete, and modify functions of user identity management can be set as flexible, multi-step workflows. Each business can customize its own approval, setup, and management processes and have multiple processes for different kinds of users.

Detailed Description Text (13):
Multi-level delegation features also simplify individual user management. Companies can assign the responsibility for maintaining user identity data to the people closest to it. For example, individual users can be allowed to: (1) add themselves to the user directory by filling out customized forms, (2) modify personal or professional information about themselves (such as addresses, personal preferences, or name changes), (3) change a piece of information in their identity profiles that can determine their access rights, or (4) allow someone else to log in as their temporary substitute while they are out of the office or on vacation. Likewise, any number of delegated administrators (both inside and outside the company) can be given the authority to: (1) create and delete users in the user directory, (2) approve a change that a user has requested, and (3) change the information about users to grant or revoke services. An administrator can be delegated any allowed degree of responsibility. For example, a company might decide that only IT staff can assign application access, whereas department managers can add new users.

Detailed Description Text (14):
External legacy systems--such as human resource management systems--can be allowed to trigger automated workflows. With this feature, a new user could be created, a departing employee could be deleted, or certain services could be granted or revoked following an event change in an external system.

Detailed Description Text (16):
Group Manager 44 allows entities to create, delete and manage groups of users who need identical access privileges to a specific resource or set of resources. Managing and controlling privileges for a group of related people--rather than handling their needs individually--yield valuable economies of scale. Group Manager 44 meets a wide range of e-business needs: easy creation, maintenance, and deletion of permanent and ad hoc groups of users who may be allowed or denied access to particular resources; modification and adaptation

of groups and their access privileges with minimal disruption to the directory server's underlying schema; efficient addition and deletion of users from established groups; and delegation of administrative responsibility for group membership and subscription requests and approvals.

Detailed Description Text (17):
With Group Manager 44, companies (or other entities) can allow individual users to do the following: (1) self-subscribe to and unsubscribe from groups, (2) view the groups that they are eligible to join or have joined, and (3) request subscription to groups that have access to the applications they need. Multi-step workflows can then define which users must obtain approval before being added to a group and which can be added instantly. Group Manager 44 also lets companies form dynamic groups specified by an LDAP filter. The ability to create and use dynamic groups is extremely valuable because it eliminates the administrative headache of continually keeping individual, static membership up-to-date. With dynamic group management features, users can be automatically added or removed if they meet the criteria specified by the LDAP filter. Dynamic groups also greatly enhance security since changes in user identities that disqualify someone from membership in a group are automatically reflected in the dynamic group membership.

Detailed Description Text (18):
The third application in the Identity System, Organization Manager 46, streamlines the management of large numbers of organizations within an e-business network, including partners, suppliers, or even major internal organizations such as sales offices and business units. Certain infrastructure security and management operations are best handled--or can only be handled--at the highest organizational unit level rather than at the individual or group level. Like User Manager and Group Manager, this application relies on multi-step workflow and delegation capabilities. Organization Manager handles the following administrative tasks: (1) organization lifecycle management, whereby companies can create, register, and delete organizations in their systems using customizable workflows; (2) maintenance of organization profiles on an attribute-by-attribute basis through self-service, delegated administration and system-initiated activities; (3) organization self-registration, whereby organizations such as business partners, customers and suppliers can self-generate a request to be added to the e-business network; and (4) creation of reusable rules and processes through multi-step workflows.

Detailed Description Text (36):
Database clients interact with database manager 120 to accomplish any database operation. Database manager 120, in turn, interacts with the profiles to determine which data stores can service the database operation. A database proxy 154 is created to service a particular database request. Database proxy 154 communicates directly to the Agents for the data stores that can service the request. The database client then interacts directly with proxy 154 to access the appropriate data stores. Thus, database proxy 154 is a dynamic object which database manager 120 creates every time a database request is made.

Detailed Description Text (38):
FIG. 3 shows database proxy 154 in dotted lines to indicate that it is created for a particular request. When the request is completed, the proxy is terminated. The proxy communicates directly with the appropriate agents for accessing the appropriate data stores. FIG. 3 shows one example of a database proxy being created to access data in data stores 36a and 36b.

Detailed Description Text (40):
BaseDB 152 calls database manager 120 in step 172, indicating the operation and search base for the data operation. In step 174, database manager 120 consults each of the profiles to determine which data store can support the operation. That is, each data store is a particular type of data store, has its own set of operations that it supports, and has its own search base that it supports. In step 176, each of the profiles indicates whether they can service the request based on whether the above mentioned criteria match the request. In step 178, database manager 120 creates proxy 154. Note that proxy 154 is for this one particular request and will be terminated at the end of the request.

Detailed Description Text (53):
FIG. 7 depicts an example of UidCookie 360. A cookie is information that a web page, system or resource stores on a client device. In some embodiments it can represent information about the user, regardless of where it is stored and in what format. This cookie includes at least three components: Uid 362, IP address 364 and timestamp 366. Uid 362 stores the user identification for the entity trying to access the Identity System. IP address 364 is the IP address of the machine that the user is currently using. Timestamp 366 indicates the time that the cookie was initially created. Some embodiments use timestamp 366 to limit the life of the cookie. Some embodiments do not use timestamp 366. In one embodiment, the cookie is encrypted.

Detailed Description Text (54):
If, in step 304, it is determined that a valid UidCookie exists, then, in step 306, the user is given access to the Identity System application requested. The Uid from the cookie is used as the user identification upon entering the Identity System. If the valid UidCookie does not exist (step 304), then it is determined whether a user identification was received in a header variable. In one embodiment using an integrated Access and Identity System, a user's request to access the Identity System will be authenticated and authorized by the Access System. After authentication and/or authorization, the HTTP request will be redirected to the Identity System. This redirected HTTP request will include a header variable labeled as "userAuth." The data associated with this header variable will indicate the user identification for the user. If the user identification was in a header variable then a UidCookie is created in step 310 and that user identification is added to the UidCookie. Subsequent to step 310, the user is provided access to the Identity System in step 306.

Detailed Description Text (55):
If the user identification was not in a header variable, then the system attempts to authenticate the user in step 312. That is, the user's user name and password provided by the login page are used to access Directory Server 36 in order to authenticate the user. More information about authentication is described below. If the user is properly authenticated, then a UidCookie is created in step 316. During an authentication process, the user's ID and password were used to access the user's identity profile in Directory Server 36. That identity profile will include a user identification, which is added to the UidCookie in step 316. In one embodiment, the user identification is the user's distinguished name. In step 318, the user is provided access to the Identity System. If the user was not properly authenticated, then the user is denied access to the Identity System in step 320.

Detailed Description Text (57):
FIG. 8 graphically depicts the various services provided by User Manager 42. Each of these services can be accessed from the User Manager home page. For example, in one embodiment, the home page will include an application selector 402, search tool 404, My Identity tab 406, Create User Identity tab 408, Deactivate User Identity tab 410, Substitute Rights tab 412, Requests tab 414 and Configure tab 416. Application selector 402 lets the user change applications from the User Manager to either the Group Manager, Object Manager or Publisher. In one embodiment, application selector 402 is a drop down menu. Search tool 404 enables a user to provide search information in order to search the directory for a set of one or more user identity profiles.

Detailed Description Text (58):
By selecting My Identity tab 406, a user is provided with the information stored in that user's identity profile. Create User Identity tab 408 allows a user with the appropriate privileges to create a new user identity profile (e.g. with a workflow). Deactivate User Identity tab 410 allows a user with proper privileges to remove an identity profile from the directory. Substitute Rights tab 412 allows the user to indicate who can proxy that user and allows the user to be a proxy for someone else. Request tab 414 allows a user to monitor workflows that are in progress or recently completed. Depending on the user's privileges, by selecting request tab 414, the user can see all workflows that involve that user, that are started by that user, that affect that user or that the user has privileges to view. Request tab 414 will indicate workflows for which there is an outstanding action to be done by the current user. The user can select that workflow and perform the task.

Detailed Description Text (60):
FIG. 9 depicts the various services provided by Group Manager 44. Once an entity is at the Group Manager home page, the entity can access the application selector 430, search tool 432, My Groups tab 434, Create Groups tab 436, Request tab 438 and Configure tab 440. My Groups tab 434 indicates the groups of which the entity is a member. By selecting any of the groups identified by My Groups tab 434 or Search Tool 432, the user will be provided with the identity profile page for that particular group. From the profile page, the group can be modified or deleted. Create groups tab 436 allows the user to create a new group. Request tab 438 provides the user with access to currently pending and recently finished workflows that involve groups. Configure tab 440 allows the user to configure various information about groups in the Group Manager. While viewing the identity profile for a group, the entity can modify that profile if the entity has appropriate privileges.

Detailed Description Text (61):
Configure tab 440 allows an entity to provide attribute access control, delegate rights, define workflows and expand dynamic groups. Attribute access control includes controlling who has view and modify permissions for each attribute in group identity profiles. Additionally, e-mail notification lists can be created which are used to notify entities when a change to an attribute is requested. Administration tasks can be delegated to local administrators. An entity can choose what rights to delegate, who to delegate to, and what the scope of the delegation is. Workflow definition includes defining the workflows for a particular group. This includes defining who is responsible for the workflow actions and who will be receiving notifications for workflow actions. Expanding dynamic groups will be discussed below. Note that some of the tabs and services may not be available to all entities, depending upon the privileges of those entities.

Detailed Description Text (62):
FIG. 10 depicts the services provided by Organization Manager 46. Organization manager 46 provides functionality to create, modify, delete and manage organizational objects. From the home page for Organization Manager 46, a user is provided with an application selector 442, search tool 444, Create Organizational Profile tab 446, Request tab 448 and Configure tab 450. Application selector 442 allows the user to select a different application to access. Search tool 444 provides a user with the ability to enter search terms in order to search for one or more organizational objects. After performing a search, the user will be provided with a list of organizational objects meeting the search requirements. User can select any of these objects to view, modify or delete, if the user has sufficient privileges.

Detailed Description Text (63):
Create Organizational Profile tab 446 allows a user to create new organizational objects, if the user has sufficient privileges. Request tab 448 allows a user to access pending workflows and workflows that have recently been finished that relate to organizational objects. Access to Request tab 448 can be restricted and/or limited depending upon users privileges. If a user has a step to perform for a workflow, it will be indicated by Request tab 448.

Detailed Description Text (64):
Configure tab 450 allows the entity to perform attribute access control, delegate administration, define workflows and define container limits. Attribute access control includes controlling who has view and modify permissions for each attribute of an organizational identity profile. In addition, an entity can specify an e-mail notification list when a change to an attribute is requested. Delegating administration includes delegating administrative tasks to local administrators. An entity can choose what rights to delegate, whom to delegate to, and the scope of the delegation. Workflow definition includes defining the workflows for a particular organization, including who will be responsible for the workflow actions and who will be receiving notifications for the workflow. Container limits includes controlling how many objects can be created in an organization. This would also include defining who will be receiving notifications that a container limit has been met, has been violated or is close to being met.

Detailed Description Text (71):
As discussed above, when an entity logs into the Identity System, the entity indicates the

entity's role. There are at least six roles: System Administrator, Master Identity Administrator, Master Access Administrator, Delegated Access Administrator, Delegated Identity Administrator and End User. The System Administrator can perform all Access System configuration tasks and all Identity System configuration tasks. The Master Identity Administrator can configure access controls, attribute access controls, new user services, workflow definitions, setting the search base, delegating rights, expanding dynamic groups, and setting container limits. The Master Access Administrator can configure a web gate, configure an access server, create host identifiers, configure users, set-up policies and policy domains, and delegate rights. The Delegated Identity Administrator is an administrator who has been delegated rights from the Master Identity Administrator. The Delegated Access Administrator can be delegated rights from a Master Access Administrator. An End User cannot perform configuration functions. There can also be a delegated admin who can create/delete users, add/remove users to/from groups, process workflow steps, etc.

Detailed Description Text (76):
FIG. 15 is a flowchart describing a process that is performed when a user becomes a proxy for another. In step 660, the system receives a request from a user to become a proxy. In one embodiment, that includes a user selecting Substitute Rights tab 412. In that tab, the system displays a list of all those persons for whom the user can be a proxy. Next to each name will be a check box. In step 662, the user selects the one person for which the user wants to be a proxy (hereinafter referred to as "the person being proxied"). For example, person A accesses Substitute Rights tab 412 to be a proxy for person B, while person B is on vacation. Person B is the person being proxied. In step 664, the user enacts the proxy right. In one embodiment, step 664 includes selecting an "enact" button. When the user selects the "enact" button, the system creates a new cookie on the users' machine called originalUidCookie. The onginalUidCookie is in the same format as the UidCookie depicted in FIG. 7. In one embodiment, the originalUidCookie is an exact copy of the UidCookie currently on the user's machine.

Detailed Description Text (80):
A lot of the tasks that are performed in the Identity System are accomplished using workflows. A workflow is a predefined set of steps that perform a specific task, where information or tasks are passed between participants and programs according to a defined set of rules. One embodiment of the present invention supports the following types of workflows: create object; delete object; change the value of attributes; and certificate issuance, revocation and renewal. In one embodiment of the present invention, a user is required to create a workflow to create or delete an object, change the value of an attribute or implement certificates. Workflows ensure that an organization's guidelines for performing a task are met. Workflows can be defined in the User Manager, Group Manager or Organization Manager. A workflow can be used only in the application (e.g. User Manager) in which it was created. Each workflow has two or more steps, including one to start the action and one to implement or commit it. Each step can contain an action, send e-mail notifications to selected persons and start the next step if its entry conditions are satisfied. A workflow is associated with a portion of the directory tree. This allows an entity to have its organizations and partners enforce different workflows. Workflows can be stored in Directory Server 36.

Detailed Description Text (84):
Workflows are created based on templates (forms) by users with sufficient privileges. In one embodiment, each template has at least four sections including a section for creating objects, deleting objects, changing attributes and working with certificates. The template provides parameters that define how workflows can be created. Templates can be edited in order to tailor the workflow definition processes. The User Manager, Group Manager and Organization Managers each have their own template files and use those template files to control and define the workflow definition process. In one embodiment, the template file is an XML document that defines a set of parameters for each of the actions available to that particular workflow type. Table 4 describes the various parameters that are used in the template files:

Detailed Description Text (86):
The first line labeled (a) indicates the workflow type, which includes creating an object, deleting an object, changing attribute or certificates. Lines (b-u) define the parameters for one particular action. One or more of the parameters described above are defined in lines (b-u)

for one action. For example, line (d) puts a value into the occurrence parameter, line (e) provides a value for a useraction and line (f) provides a value for forceCommit. Lines (h-k) provide the various pre-actions that have to occur before the particular action is performed. Lines (l-o) provide exit conditions. While the generic template above shows line (b-u) for one particular action, a typical template would have parameters for many actions. One template is likely to be used to create many workflows.

Detailed Description Text (87):
FIG. 16 is a flowchart providing an overview of the process for creating a workflow. In step 700 a template is created and stored. In one embodiment, the template can be created using a word processor. In step 702, a workflow object is created. The workflow can be created using the User Manager 42, Group Manager 44 or Organization Manager 46. In step 704, the steps of the workflow are defined based on the template created in step 700. In step 706, the workflow is stored. In step 708, the workflow is performed. Additional workflows can be created by performing steps 702-708 because once a template is created, it can be used to create many workflows.

Detailed Description Text (88):
FIG. 17 is a flowchart describing the steps of creating a template. In step 730, each workflow type is added to the template file. In reference to the generic template above, line (a) of the generic template identified the first workflow type. It is likely that the workflow types would include create object, delete object, change attributes and certificates. In step 732, for each workflow type, actions are added. Code for one action is depicted above in the generic template. In step 734, for each action the parameters are added. In one embodiment, one or more domains can be specified for a template or for workflow types in the template. If domains are specified, then the associated template or workflow types only apply to workflows created for the specified domain(s).

Detailed Description Text (89):
FIG. 18 provides a flowchart for creating a workflow object (step 702 of FIG. 16). In step 750, the appropriate manager (User, Group, Organization) receives a selection or indication to create a workflow. In step 754, it is determined whether the user is allowed to create the workflow. If no, the process is completed. If yes, the system identifies the different types of workflows, objects, tasks and target domains for which the user can create a workflow (step 756). In step 758, the user selects the identification of the workflow to be created. The identification is just a unique name to identify the workflow. In step 760, the user inputs a selection of the type of workflow based on the options from step 756. Step 760 includes choosing the task that the workflow will perform. For example, in the User Manager, the possible tasks include create a user, delete a user, change attribute, etc., as discussed above.

Detailed Description Text (91):
FIG. 19 is a flowchart describing the process of defining steps for a workflow being created. The process of FIG. 19 is performed based on the template. In step 780, the system determines the possible actions that can be performed for this particular workflow based on the template. That is, the system reads the template and determines which actions can be added. The actions that can be selected are added to a GUI in step 782 and, in step 784, a selection from the GUI is made by the user. In step 786, the system determines which types (required, optional, supplied) of attributes are available, based on the template. The appropriate attributes and types of attributes are added to the GUI in step 788. For example, the various attributes can be selected as required, optional or supplied. If the template does not allow for the supplied attribute, then that option will not be available on the GUI.

Detailed Description Text (95):
A subflow is a workflow that is initiated by another workflow. The concept of subflow was introduced and implemented to reduce administrative work. If a workflow already exists to perform a task, any other workflow that needs to perform that task should be able to leverage off the first workflow. When creating a workflow, an indication that there is a sub-workflow is provided by the creator of the workflow when the creator indicates that one or more of the variables are supplied.

Detailed Description Text (96):
The workflow that initiates the subflow is referred to as the parent workflow. A workflow can be both a parent workflow to a first workflow and a subflow to a second workflow. The parent workflow may or may not wait for the subflow, as defined in the workflow creation. Consider the following example, a company uses a first workflow to create new users for the Identity System and add the new user's identity profile to the directory. As part of its process, the new user workflow obtains the new user's telephone number. The obtaining of the new user's telephone number is accomplished by performing a new telephone number workflow. In this example, the new telephone number workflow is initiated by a step in the new user workflow. Therefore, the new telephone number workflow is a subflow of the new user workflow. In one alternative, the new telephone number workflow can also call a subflow, for example, to get a new telephone line connected and operational. This, second subflow can also call a subflow, and so on. There can be many levels of nesting of subflows. Additionally, a parent workflow can have many subflows.

Detailed Description Text (98):
FIG. 20 is a flowchart describing the process of using a workflow. The process of FIG. 20 is performed, for example, when creating a new user, a new group, etc. In step 840, the relevant manager (e.g. user, group or organization) receives a request to perform an action that requires a workflow. Most actions are likely to have an effect on at least one identity profile in the directory. In step 842, it is determined whether this user is allowed to initiate the workflow. If not, the process of FIG. 20 is completed. If so, the GUI determines and reports a set of one or more workflows. This set of one or more workflows meets three criteria: (1) the user is allowed to use the workflows, (2) the workflows perform the requested task and (3) the workflows are associated with a domain that includes the target of the task. For example, if user A has requested to modify the attributes of Employee 8 (identity profile 264 of FIG. 5), then the system will identify and report workflows that (1) user A has permission to access, (2) perform attribute modification and (3) are associated with a domain that includes identity profile 264 of FIG. 5. In one embodiment, the identified workflows are displayed in a menu.

Detailed Description Text (99):
In some situations, a workflow is requested without knowing the location of the target identity profile. For example, a user can request to create an object without indicating where to store the object in the directory. In such a scenario, the system will find and report workflows that perform the requested task and can be accessed by the user. When the system reports the list of workflows (e.g. via a GUI), the system will also report the domain associated with each workflow. In this situation, step 846 includes the system receiving a selection from the user of the workflow desired, and the domain to operate on.

Detailed Description Text (101):
In step 854, pre-notifications, if any, defined in the workflow are sent out. In step 856, cross application workflows, if any, are invoked, as per the event catalog. In step 858, the current step of the workflow is performed. In step 860, it is determined whether there are supplied variables. When creating a workflow, the creator had the option of defining the types of variables. Supplied variables are those variables whose value will be supplied by a subflow. If the current step has a supplied variables, then the system searches for any workflows that can supply the variable and apply to the appropriate domain. If only one workflow is found for each supplied variable, then those workflows are initiated as a subflow in step 862. If multiple workflows are found for a particular supplied variable, then the user is given a choice and the chosen workflow is initiated as a subflow in step 862. Note that the subflow could itself have a subflow, which could itself have a subflow, and so on. There is no limitation on the number of subflow nestings.

Detailed Description Text (103):
In step 874, the system determines whether the user is allowed to perform the next step. If not, the process of FIG. 20 is stopped. If so, the system determines in step 876 whether it has to wait for the subflow(s) started in the previous workflow step. In one embodiment, a flag is set at workflow creation time to indicate that the workflow should wait or not wait. If there is a subflow and the current workflow has to wait, the system continues to wait until the subflow is completed. If there is no subflow or it does not have to wait, then the system

determines whether all entry conditions have been satisfied in step 878. If not, the system waits for the entry conditions to be satisfied. If yes, the process continues to step 856.

Detailed Description Text (107):
A workflow is performed by one of the three managers described above (User Manager, Group Manager, Organization Manager). There may be cases when one workflow in one of the applications (e.g. user manger) needs to trigger a workflow in another application (e.g. Group Manager). For example, when creating a new user with a workflow in the User Manager, it may be beneficial for that workflow to trigger another workflow in the Group Manager which subscribes the new user to groups. A cross application workflow is performed using the event catalog described above, a client program and (optionally) a configuration file for the client program, all of which will be described below.

Detailed Description Text (109):
Actions are functions or applications that perform a task in response to an event. These actions are defined to enhance the base functionality of the system of FIG. 1. Multiple actions can be defined for each event. Actions are executed in the order that they appear in an event catalog. Actions are defined using a plug-in model similar to Web Server CGI model. Functions are applications defined for each custom action. Each function/application will take a standard XML structure as its parameters that allow the system to specify information about the event that triggered the function. Action functions are defined within libraries (.dll or so) or stand alone executable files. To create a new action based on an event, one must insert a hook into the event catalog. All entries in the event catalog are defined in the following format:

Detailed Description Text (115):
FIG. 22 is a flowchart describing the process for creating a cross application workflow situation. In step 940, the first workflow for the first application is created. For example, the create user workflow for the User Manager application is created. In step 942, the second workflow for the second application is created. For example, the subscribe user to group workflow can be created in the Group Manager application. In step 944, an entry is added to the event catalog. For example, an entry is added to the event catalog that indicates the workflow ID for the workflow created in step 940--the step that should spawn the second workflow and that it is a post event. The entry also identifies the client program that will be created (see below). In step 946, the client program is created which invokes the second workflow. This client program receives the distinguished name of the newly created user as a parameter. In another embodiment, the client program receives other attributes from the identity profile being operated in by original workflow. The client also receives the workflow instance, the work step identification, and attributes of the work step. In step 948, a configuration file may be created for the client program. For example, if the second workflow is to subscribe a user to a group, then the configuration file may include rules for which users should be added to which groups.

Detailed Description Text (143):
Another feature of Group Manager 44 is the ability to perform group expansion. Expanding a group means evaluating the LDAP rule that specifies its dynamic membership and then updating the static membership list with results of the evaluation of the LDAP rules. Expansion, in effect, populates the static membership with a snapshot of the dynamic membership at the time of expansion. Expansion has performance implications. On one hand, it is much faster to evaluate group membership by looking up a value in the static membership list than to evaluate the rule that specifies dynamic membership. On the other hand, frequently updating groups is, in itself, computationally expensive. If the expansion occurs as a separate process, the performance hit can be hidden from the user. Thus, if a group is already expanded when a user requests to see all the members of a group, the processes of FIG. 27 or 28 do not need to be performed again because the group only has static members at this point. An administrator should expand the groups regularly to maintain accuracy. It may be possible to create a background process that automatically expands a group at certain intervals.

Detailed Description Text (145):
When an entity accesses the groups that were expanded in step 1354 and/or requests to see the members of the group (step 1356), the entity sees the expanded list of members. Additionally,

any process that needs to access members of a group will access the membership generated in the expansion process. In one embodiment, the process of FIG. 31 can be automatically repeated (step 1358) using a background process or any other means.

Detailed Description Text (147):
Another feature of Group Manager 44 is the ability to dynamically modify groups during run time. This feature is based on attaching auxiliary object classes to structural object classes. A structural object class can be instantiated to create a group such that for each entry in the directory there is only one structural object class. The structural object class cannot change after the object has been instantiated and is being used. One or more auxiliary object classes can be attached to any structural object class in a directory. The structural object class defines a set of attributes. The auxiliary object class also has a set of attributes. When an auxiliary object class is attached to an object class, the attributes of the auxiliary class are added to the object. Once instantiated, a structural object class cannot be modified or removed; auxiliary object classes, however, can be added or removed. Group manager 44 provides the user with the ability to add or remove auxiliary object classes on the fly using a GUI.

Detailed Description Text (148):
Prior identity systems allow for the addition of auxiliary classes to structural classes upon creation of the object. The present invention allows for auxiliary classes to be added and removed subsequent to object creation. That is, dynamically, an existing object class can have additional attributes added to the group object or removed from the group object by adding or removing auxiliary classes.

Detailed Description Text (149):
When creating a group, an administrator (or other user with sufficient privileges) is provided with a graphical user interface that lists all possible attributes that can be included in the group profile. Some of these attributes are part of structural object class, while others are part of auxiliary object classes (or auxiliary object class schema). If the user selects attributes from an auxiliary class, then those auxiliary classes are added to the object upon creation of the object. After the group is created, various attributes can be populated with data values. Subsequent to this time, attributes that are associated with auxiliary classes can be removed or added to the group. In addition to adding flexibility to defining which attributes are associated with a group, the present invention allows for bulk deletion of attributes. Simply removing the auxiliary object class from the group entry will automatically delete all attributes of the removed auxiliary object class.

Detailed Description Text (150):
FIG. 32 is a flowchart describing an overview of the process for adding and removing attributes to a group during run time. In step 1398, a group is created. This step includes determining which attributes to include in the group definition. Based on the attributes chosen, a structural class and the appropriate auxiliary classes are added to the group. In one implementation, the group is created by instantiating the appropriate classes to create a group object representing the group identity profile. In one embodiment, a group can be created that has an auxiliary class, but no attributes of that auxiliary class. The system can use a workflow to create the group and the workflow knows which auxiliary classes to use. The arrow from step 1398 to step 1400 is depicted as a doted line to indicate that time and other steps pass before step 1400 is performed. That is, step 1400 is performed after a group has been created and, possibly, after the various attributes have been populated with data. In step 1400, Group Manager 44 receives a request to modify the existing group. This can happen from Configure tab 440. Alternatively, while viewing a group, Group Manager 44 will display a "modify group" button. Selecting that button allows the user to request a modification to the group being viewed, if the user has sufficient privileges. In step 1402, Group Manager 44 provides a list of auxiliary classes that can be added or removed from the existing group. In an alternative embodiment, Group Manager 44 provides a list of attributes to add or remove, with each of the attributes being associated with auxiliary classes. The auxiliary classes and/or attributes to be added or removed are reported to the user via a graphical user interface. Next to each class (or each attribute) is a check box. The user can check the check box to indicate that the class (or attribute) should be added. The user can uncheck check box to indicate that the class (or attribute) should be removed. In step 1404, the selection of

classes (or attributes) to be added and removed are received by Group Manager 44 from the graphical user interface and stored. In step 1406, those auxiliary classes selected to be removed are then removed from the group object including removing those attributes from the group object. In step 1408, the auxiliary class selected to be added and their associated attributes are added to the group object. After step 1408, the group can be used as any other group; for example, a user can be authorized to access a resource based on attributes of or membership in a group.

Detailed Description Text (157):
The following two attributes belong to neither of the two categories above but are included here for completion: obgroupsimplifiedaccesscontrol--stores the initial attribute access control policy applied to newly created group obgroupadministrator--stores the user selected as the group administrator.

Detailed Description Text (158):
The system of FIG. 1 provides users with a variety of interface options. For example, the system supports users with traditional browsers by providing for communication using HTTP and Hypertext Mark-up Language ("HTML"). The system also supports interfaces to third party applications, proprietary browsers and others by providing for communication using Extensible Mark-up Language ("XML"). Embodiments of the present invention provide further flexibility by facilitating the use of custom XML templates to generate HTML and XML responses.

Detailed Description Text (186):
The use of templates and stylesheets provides users with a great deal of flexibility and control. Templates and stylesheets can be modified to address the unique needs of system users. Different system users employing the same programs can create different displays of the program's results. Users and/or system administrators implement customized templates and stylesheets in desired register files.

Detailed Description Text (196):
In one embodiment, Identity Server 40 obtains attribute display characteristics from directory entries in Directory Server 36. Each Directory Server entry corresponds to a different attribute type. For each attribute, Identity Server 40 locates a corresponding directory entry, which provides the attribute's display characteristics. In one such embodiment, a system administrator creates all the display attribute directory entries when Identity System 40 is configured. In alternate embodiments of the present invention, the directory entries are replaced by tables, data structures, or other means that relate display characteristics to attributes so the display characteristics can be obtained by Identity Server 40.

Detailed Description Text (199):
By employing post-processing, a client can create a plug-in program running on Identity Server 40 that captures and modifies the Output XML prior to Identity Server 40 returning a request response. This provides users of Identity Server 40 with great flexibility and control over the content and format of request responses. For example, a user can modify the Output XML to insert a customized display type directive or remove unwanted data.

Detailed Description Text (202):
One of the peripheral programs frequently performed in conjunction with a user's expressly requested program is the generation of a navigation bar. The navigation bar is displayed along with the result of a user's expressly identified program--enabling the user to navigate within the request response and other related areas. For example, the navigation bar lets a user scroll through the text of the response and jump to related data in Directory Server 36. Some implementations of Identity Server 40, however, provide users with different levels of access to Directory Server 36 and functions performed by Identity Server 40. Identity Server 40 provides for displaying different navigation bars based on user access privileges.

Detailed Description Text (209):
If the attempted data store access is a write (step 1834), Identity Server 40 determines whether the requested memory location is stored in cache object 1829 (step 1844). If the entry is stored in cache object 1829, Identity Server 40 removes the old entry in cache object 1829

(step 1846) and writes the data supplied by the request into cache object 1829 (step 1845). Identity Server 40 also writes the same data into the data store (step 1848). If no cache entry exists for the requested entry, Identity Server 40 creates a space for the entry in cache 1829 and writes the data to cache 1829 (step 1845). Identity Server 40 also writes the data to the data store through data store 36 (step 1848).

Detailed Description Text (236):
Certificate Processing Server 2076 obtains a digital signature for the request from signing device 2078 (step 2160). Certificate Processing Server 2076 then forwards the digitally signed request as a certificate signing request to Certificate Authority 2084 (step 2162). Certificate Authority 2084 responds by creating a certificate (step 2164) and forwarding the certificate to Certificate Processing Server 2076 (step 2166). Certificate Processing Server 2076 forwards the certificate to certificate registration module 2072 (step 2170). Certificate registration module 2072 stores the new certificate in certificate data store location 2082 (step 2156). Certificate registration module 2072 then notifies the user that the certificate is in place (step 2158).

Detailed Description Text (237):
Once a certificate has been issued it is typically valid for a predetermined period of time, such as one year. After the time period expires, the certificate holder must renew the certificate. In one embodiment of the present invention, the certificate holder renews the certificate by submitting a certificate renewal request to Identity Server 40. This request is handled by certificate registration module 2072 in essentially the same manner as described above for certificate enrollment. The same process is applicable, because the renewal of a certificate is essentially the same as enrollment. When a certificate is renewed, Certificate Authority 2084 generates a new private key-public key pair, in essence creating a new certificate without increasing the total number of certificates issued to the Identity System. The only difference is that Certificate Processing Server 2076 informs Certificate Authority 2084 that a certificate is to be renewed, as opposed to a new certificate being issued.

Detailed Description Text (249):
The cert_generate_certificate action in the enrollment workflow causes certificate registration module 2072 to obtain a certificate. As shown above, certificate registration module 2072 obtains certificates with the assistance of Certificate Processing Server 2076 and Certificate Authority 2084.

Detailed Description Text (252):
As shown above, several of the workflow actions are optional. The flexibility to add different steps to workflows makes certificate management very flexible. System administrators can create different certificate related workflows for different types of users. For example, a particular type of user may be automatically granted a certificate upon requesting enrollment--requiring the workflow to include only the cert_initiate_enroll and cert_generate_certificate actions. Alternatively, another type of user may require approval before a certificate is issued--requiring the workflow to include an approval or provide_approval action. In further embodiments, system administrators can also initiate certificate-related requests on behalf of system users.

Detailed Description Text (266):
FIG. 60 is a flow chart, which describes the process of creating a policy domain. In step 2400, the Access System receives a request to create a policy domain. In step 2402, the name of the policy domain and the description of the policy name are stored. In step 2404, one or more URL prefixes are added to the policy domain. In step 2405, one or more host ID's are added to the policy domain (optional). Next, one or more access rules are added to the policy domain (steps 2406 and 2408). An access rule is a rule about accessing a resource. Examples of access rules include authorization rules, authentication rules, auditing rules, and other rules, which are used during the process of attempting to access a resource.

Detailed Description Text (268):
After setting up the authentication rule in step 2406, one or more first level or default authorization rules are added to the policy domain in step 2408. In general, an authorization

rule determines who can access a resource. The default authorization rule allows or denies users access to resources within its applicable policy domain. If multiple authorization rules are created, then they are evaluated in an order specified in step 2410. In step 2412, a first level (default) audit rule is configured for the policy domain. In step 2414, zero or more policies are added to the policy domain. In step 2416, the data for the policy domain is stored in Directory Server 36 and appropriate caches (optional) are updated. In one embodiment, an authorization rule or an authentication rule can be set up to take no action. That is, always grant authentication without any challenge or verification; or always grant authorization without any verification.

Detailed Description Text (283):
In step 2528 of FIG. 64, the authentication rule is created in accordance with the method of FIG. 63. In step 2530, one or more authorization rules are created for the policy in accordance with the method of FIG. 61. In step 2532, an audit rule for the policy is configured. In step 2534, POST data (optional) is added to the policy. This POST data is used to map resources with policies.

Detailed Description Text (285):
In one embodiment of the present invention, an authentication scheme can specify one of four challenge methods: none, basic, form, and X.509. If an authentication scheme's challenge method is set to "none," no authentication is required to access a requested resource, thus allowing support for unauthenticated users. This challenge method can be used over both unsecured as well as SSL connections. The "basic" challenge method can also be used over both unsecured and SSL connections. The "X.509" challenge method can be used over an SSL connection between a user's browser and Web Server host. A "form" challenge method employs a custom, site-specific HTML form presented to the user, who enters information and submits the form. Subsequent processing is determined by the administrator at the time the authentication scheme is created. Form challenge methods can be used over both unsecured and SSL connections.

Detailed Description Text (305):
To increase the speed of pattern matching, the system tries to do some work up front. When Access Server 34 loads a pattern, it creates an object. This object's constructor "compiles" the pattern. This compiling is essentially building a simple state machine from one pattern to other, i.e., it creates a chain of "glob nodes." Each glob node consists of either one pattern or a node set. For example, consider pattern:

Detailed Description Text (315):
In a more complex case, an e-business host company's web presence incorporates associated web sites whose Web Servers have names in multiple domains. In such a multiple domain case, each of the associated portal Web Servers use a Web Gate plug-in configured to redirect user authentication exchanges to the e-business host's designated web log-in Web Server. The user is then authenticated at the e-business host's web log-in server, and an encrypted cookie is issued for the e-business host's domain to the user's browser. The user's browser is then redirected back to the original associated portal's site where the Web Gate creates a new cookie for the associated portal's domain and returns it to the user's browser.

Detailed Description Text (319):
FIG. 71 provides a block diagram of an authentication cookie 3150 passed by Web Gate 28 to browser 12 in step 2580 of FIG. 65. Cookie 3150 is encrypted with a symmetric cipher so that cookies from all instances of Web Gate 28 in a given deployment of the present invention may be encrypted using the same key. This key (called a shared secret) is stored on Directory Server 36 and distributed to each of the Web Gates 28 by Access Server 34. The shared secret can change as often as desired by an administrator. In one embodiment of the present invention, cookie 3150 is encrypted using RC4 encryption with a 2048 bit key. In one embodiment, previously valid keys are grandfathered such that both the current key and the immediately prior key will both work to de-crypt encrypted cookie 3150. The present invention features a one-button key re-generation function. This function is easily scriptable.

Detailed Description Text (320):
In one embodiment, the information stored by cookie 3150 includes: (1) the authentication level

3152 of the authentication scheme used to create the cookie, (2) the user ID 3154 of the authenticated user, (3) the IP address 3156 of the authenticated user, and (4) session start time 3158 identifying the time at which cookie 3150 was created. If the time elapsed since the session start time 3158 exceeds a maximum session time, the cookie will become invalid. Idle start time 3160 is also stored, which identifies the time when the previous HTTP request for a protected resource was made in which cookie 3150 was passed. If the time elapsed since the idle start time 3160 exceeds a maximum idle time, the cookie will become invalid. Both of these time limits force users to re-authenticate if they have left a session unattended for longer than the maximum session or idle times. In one embodiment, user ID 3154 stores the distinguished name for the authenticated user.

Detailed Description Text (321):
Cookie 3150 also stores a secured hash 3162 of information 3152, 3154, 3156, 3158, and 3160. In one embodiment of the present invention, secured hash 3162 is created using an MD5 hashing algorithm. Most Internet browsers cache a user's supplied authentication information during basic and certificate authentication challenge methods, and then transparently re-send the information upon receiving an authentication challenge from a Web Server. In one embodiment, an administrator can enable a form authentication challenge method requiring end users to re-authenticate upon expiration of the maximum session or maximum idle time limits.

Detailed Description Paragraph Table (2):
TABLE 1 Application Workflow Tasks User Manager Create User Delete User Change Attribute Certificate Enrollment Certificate Renewal Certificate Revocation Group Manager Create Group Delete Group Change Attribute Org. Manager Create Object Delete Object Change Attribute

Detailed Description Paragraph Table (3):
TABLE 2 Workflow Type Actions Creating object Initiate Self Registration Provide Information Approval Provide Information and Approval Activate Commit Error Report External Action Deleting object Initiate Change Information Approval Change Approval Deactivate Commit Error Report External Action Changing Attribute Request Approval Provide Information Provide Information and Approval Commit Error Report External Action

Detailed Description Paragraph Table (4):
TABLE 3 Action Description initiate This action initiates workflows. Required, option, and supplied attributes may be configured for this action. Based on the relevant data configured in the step, the action will compose a page for the user to fill in the required information and to add additional attributes for provisioning (supplied variables) if so desired. Once the page is submitted, the workflow engine will trigger the Change Attribute workflows for the supplied attributes. People who are configured as a participant for this action and its corresponding workflow will see the "Create Profile" or "Initiate Deactivate User" button. self_registration This action allows an e-user to fill in a registration form and submit it for acceptance. The required information will be displayed on the page. It is envisioned that self-registration will be used before the user has access to an application. Therefore, the UI of this page will be designed without the context of an application and with credentials for authentication. request This action makes a request for change/add/delete attribute. People who are configured as a participant for this action and its corresponding workflow will see the "Request to Modify" or "Request to Remove" button on the profile page (during "modify" mode). provide_info This action is similar to initiate, in that it collects information from the user and triggers other workflows, if necessary. It is treated as a different action from initiate for the following reasons: Initiate is always the first action in the workflow. Provide_info can occur at multiple places in a workflow while initiate can not. The people who can initiate the workflow may be different from those who can provide intermediate information. Only the people configured as the participants for the initiate action will see the "Create Profile" button. Provide_info will try to retrieve the required attributes to display the values to the user. This allows the information setup in the previous steps or in the directory to be changed. change_info This action is identical in behavior to provide_info. A different name is used because the name change_info makes more sense in the case of deactivating. approval This action can be configured with only the required attributes. At run time, the values of the required attributes will be presented to the user to get approval. No information is supposed to be changed. The only user action allowed is to click on the button to indicate approve or reject.

In other embodiments, a digital signature could be used to provide a nonrepudiation approval. Provide_info and approval This action combines the provide_info and the approval into one action. In some situation, customers may want the people who can approve also to be able to provide or change the information if necessary. change_approval This action is identical in behavior to provide_approval. A different name is given to reflect the nature of the action in the deactivating context. activate This action enables the user to explicitly mark an entry ready. Until this action is performed, the user's entry has been marked as "PendingActivation." Upon completing this action, the status will be changed to "Activated." Once "Activated," this user entry may be used for authentication to the system. deactivate This action is the counterpart of the activate action to mark an entry suspended. Until this action is performed, the user's entry has been marked as "Pending for Deactivation." Upon completing this action, the status will be changed to "Deactivated." In both of these cases, this entry will not be recognized as an authorized user in the system. commit This action writes the information collected this far in the previous workflow steps to the directory. Commit can be done multiple times. The location of the write is the user's permanent location as selected in the "initiate" step. error_report This action is to report for a background process. When a background process encounters a processing error, it has no proper way to report the error since there is no responsible person for the action. The workflow definer can configure the failed path to this error_report step, so that the error can be designated to the responsible individuals. external_action External action can be plugged into the workflow as a distinct step.

Detailed Description Paragraph Table (16):
TABLE 8 Workflow Actions Certificate Enrollment cert_initiate_enroll provide_info (optional) approval/provide_approval (optional) cert_generate_certificate Certificate Renewal cert_initiate_renew provide_info (optional) approval/provide_approval (optional) cert_generate_certificate Certificate Revocation cert_initiate_revoke cert_revoke_certificate

CLAIMS:

27. The method of claim 20, further comprising: determining that said cache does not contain said particular entry in said cache; and creating said particular entry in said cache.

Previous Doc Next Doc Go to Doc#